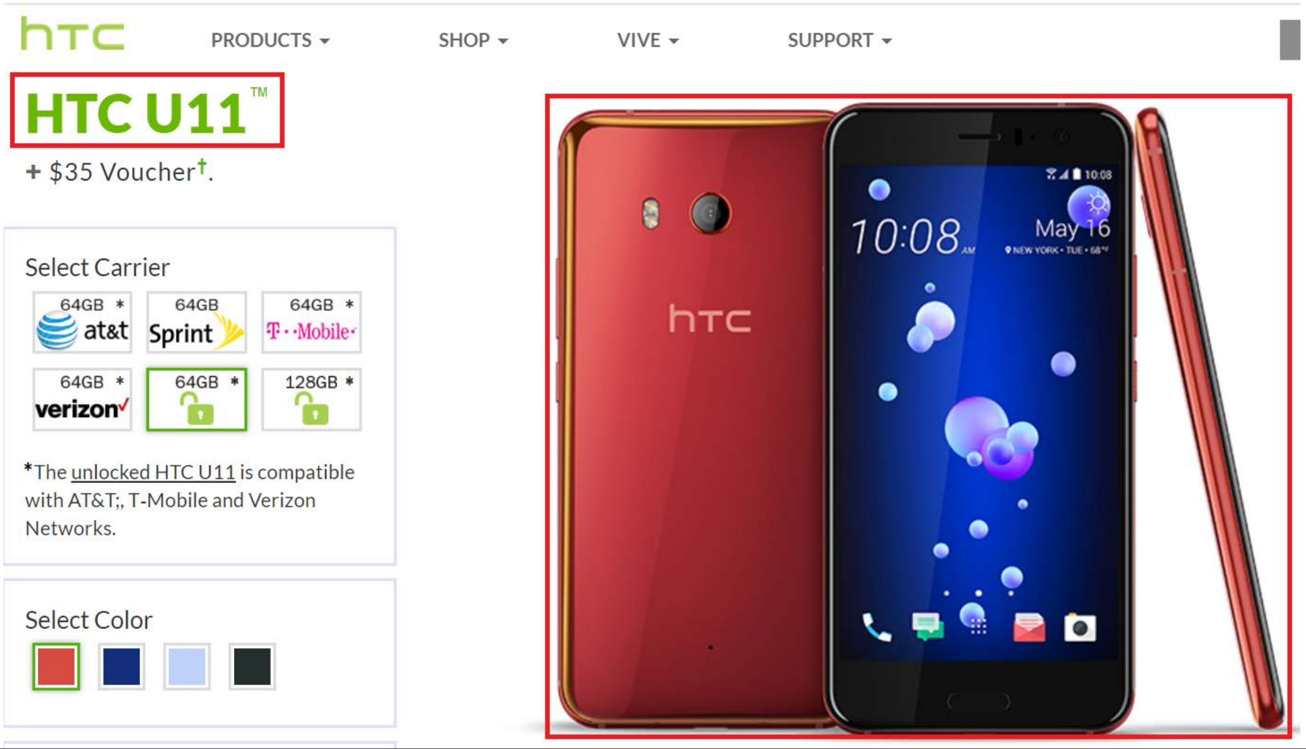
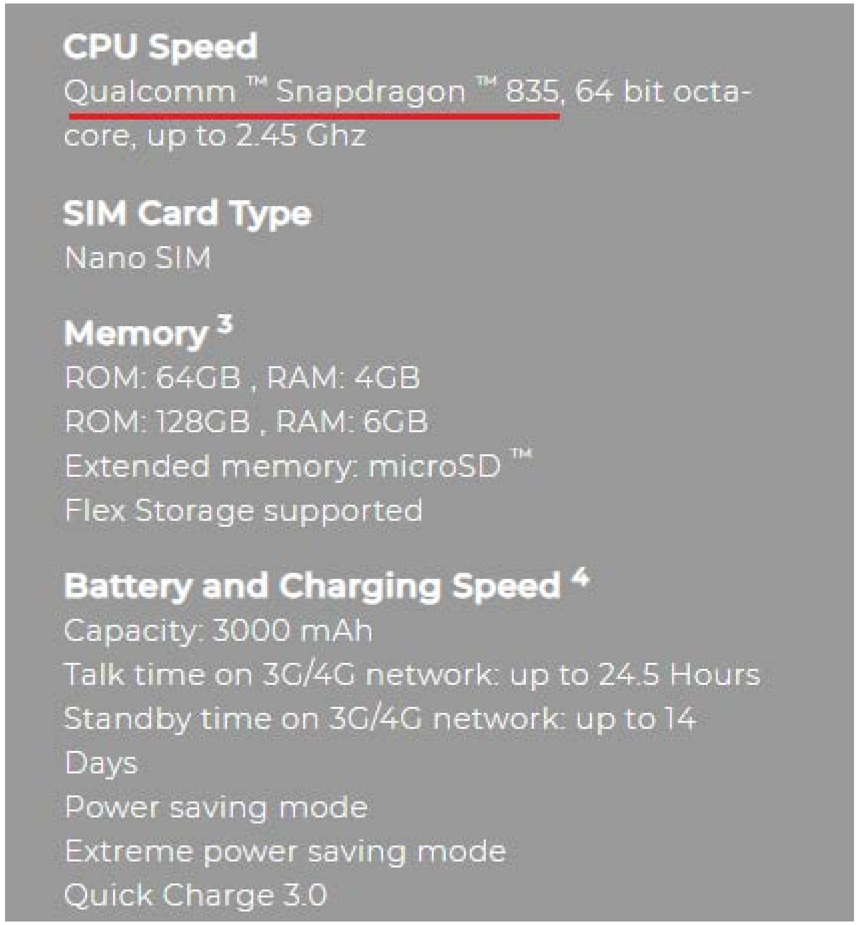
EXHIBIT B

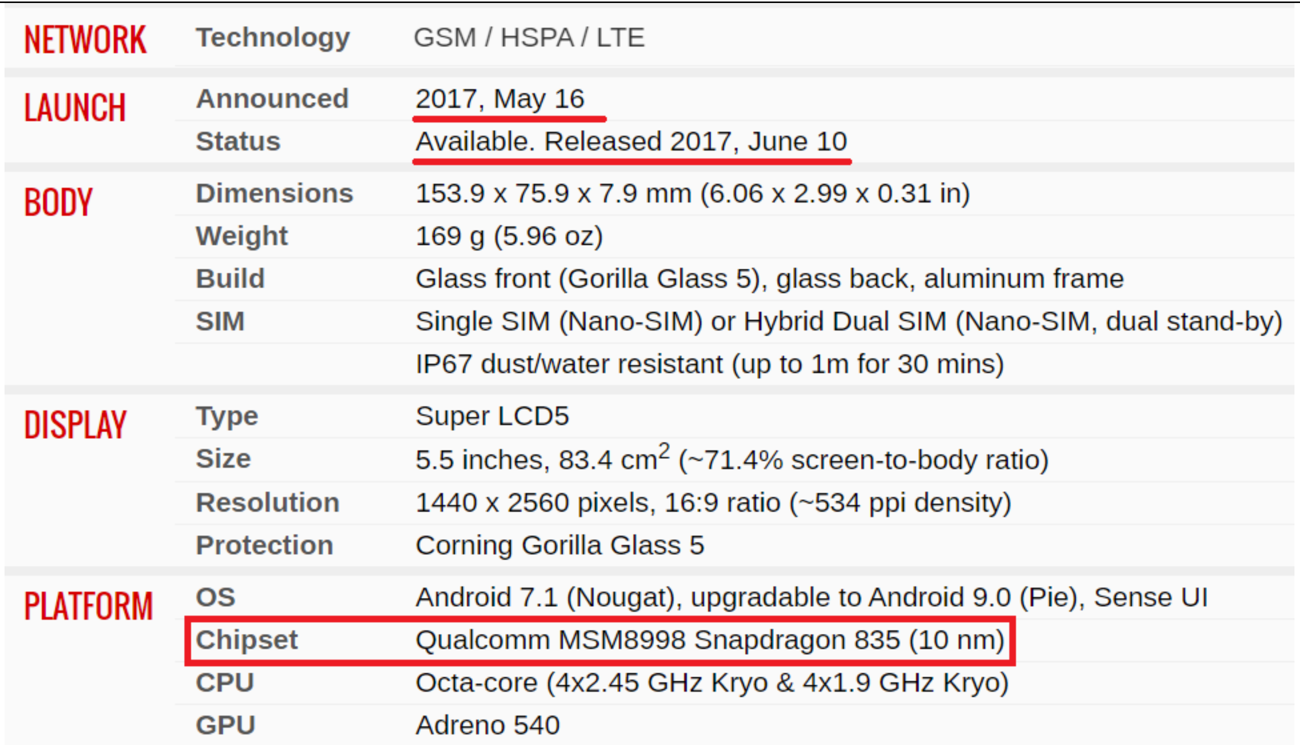
Charted Claims:

Method Claims: 1

Non-Method Claims:

US6819539	HTC U11 ("The Accused Product")
1. A method for circuit recovery from overstress conditions, comprising the steps of:	The accused product discloses a method for circuit recovery from overstress conditions (e.g., deviation of voltage from its normal values). As shown below, the HTC U11 utilizes a Qualcomm Snapdragon 835 processor.  HTC U11™ + $35 Voucher†. Select Carrier 64GB * at&t · 64GB Sprint · 64GB * T-Mobile 64GB * verizon · 64GB * · 128GB * *The unlocked HTC U11 is compatible with AT&T;, T-Mobile and Verizon Networks. Select Color 🟥 🟦 ⬜ ⬛

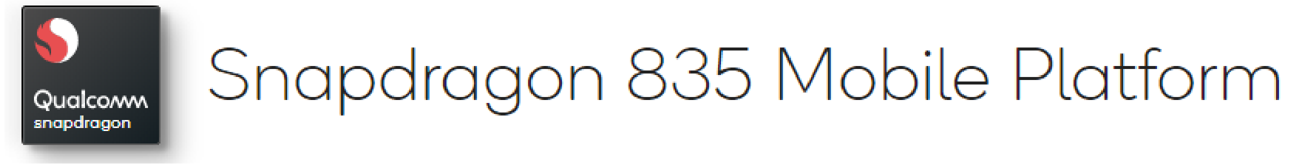
	https://web.archive.org/web/20180226131131/http://www.htc.com/us/smartphones/htc-u11/buy#!carrier=unlocked&color=red  **CPU Speed** Qualcomm ™ Snapdragon ™ 835, 64 bit octa-core, up to 2.45 Ghz **SIM Card Type** Nano SIM **Memory**[3] ROM: 64GB , RAM: 4GB ROM: 128GB , RAM: 6GB Extended memory: microSD ™ Flex Storage supported **Battery and Charging Speed**[4] Capacity: 3000 mAh Talk time on 3G/4G network: up to 24.5 Hours Standby time on 3G/4G network: up to 14 Days Power saving mode Extreme power saving mode Quick Charge 3.0 https://www.htc.com/us/smartphones/htc-u11/

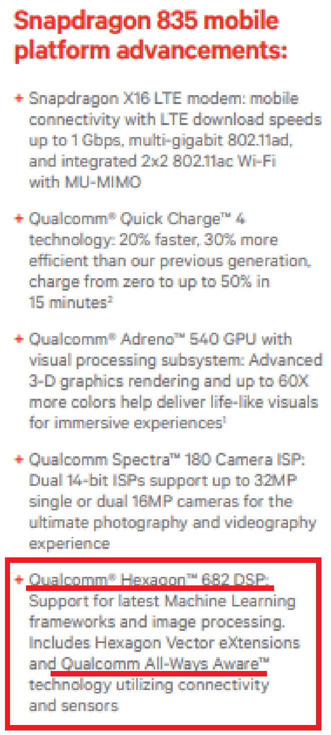
NETWORK	Technology	GSM / HSPA / LTE
LAUNCH	Announced	2017, May 16
	Status	Available. Released 2017, June 10
BODY	Dimensions	153.9 x 75.9 x 7.9 mm (6.06 x 2.99 x 0.31 in)
	Weight	169 g (5.96 oz)
	Build	Glass front (Gorilla Glass 5), glass back, aluminum frame
	SIM	Single SIM (Nano-SIM) or Hybrid Dual SIM (Nano-SIM, dual stand-by)
		IP67 dust/water resistant (up to 1m for 30 mins)
DISPLAY	Type	Super LCD5
	Size	5.5 inches, 83.4 cm^2 (~71.4% screen-to-body ratio)
	Resolution	1440 x 2560 pixels, 16:9 ratio (~534 ppi density)
	Protection	Corning Gorilla Glass 5
PLATFORM	OS	Android 7.1 (Nougat), upgradable to Android 9.0 (Pie), Sense UI
	Chipset	Qualcomm MSM8998 Snapdragon 835 (10 nm)
	CPU	Octa-core (4x2.45 GHz Kryo & 4x1.9 GHz Kryo)
	GPU	Adreno 540

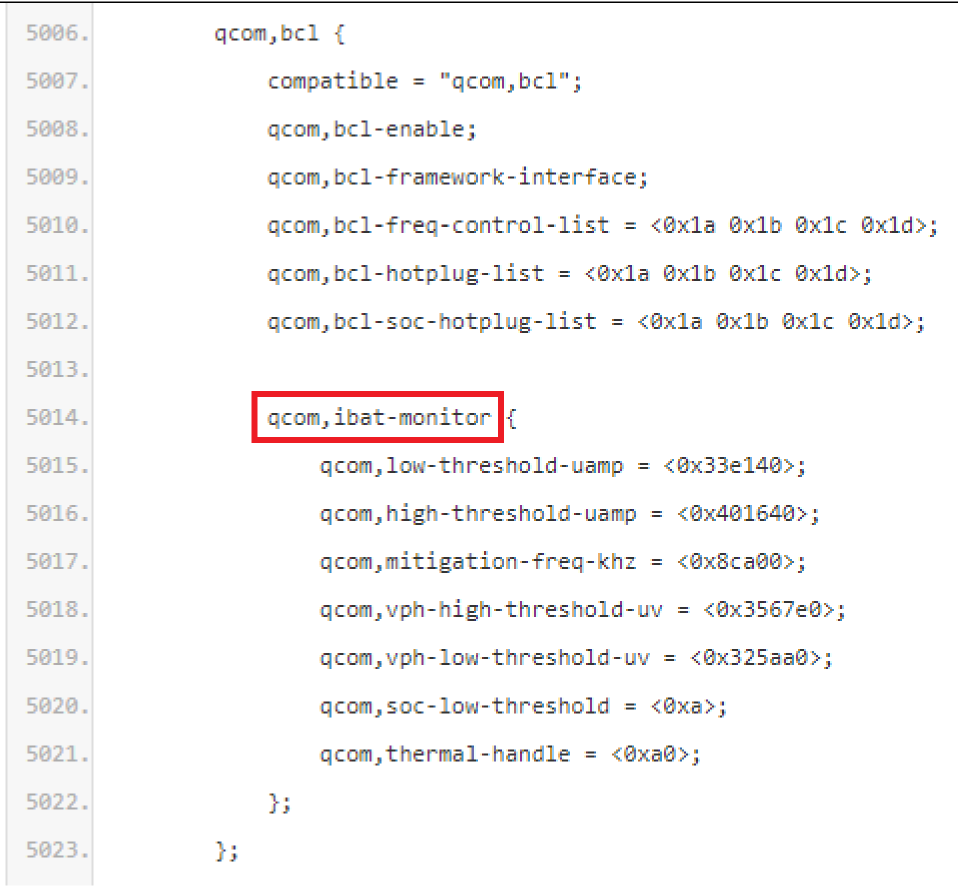
https://www.gsmarena.com/htc_u11-8630.php

As shown below, the Snapdragon 835 includes a battery monitoring circuit that generates a signal based upon the occurrence of a certain condition (in this case voltage variances for normal values).



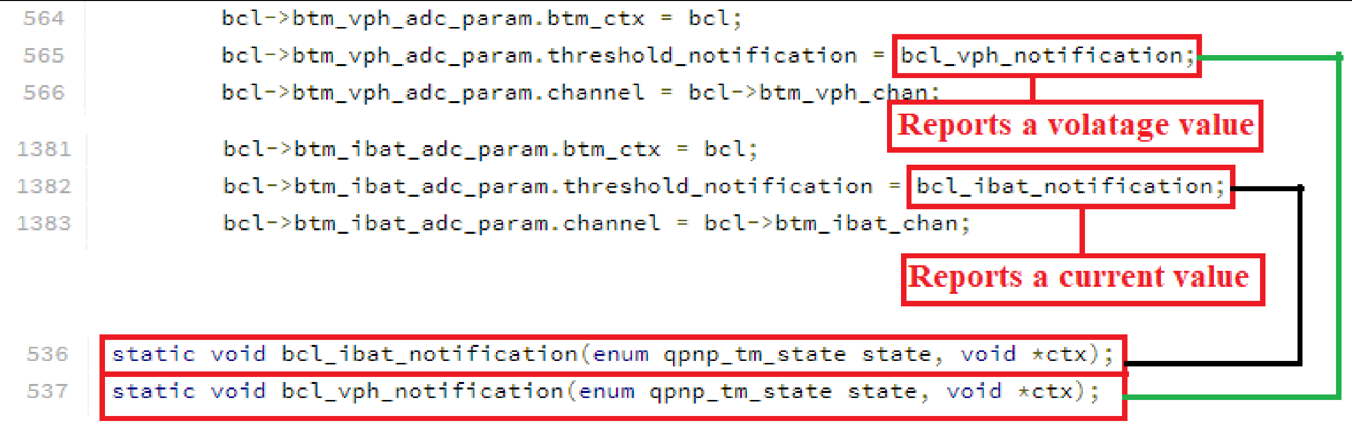
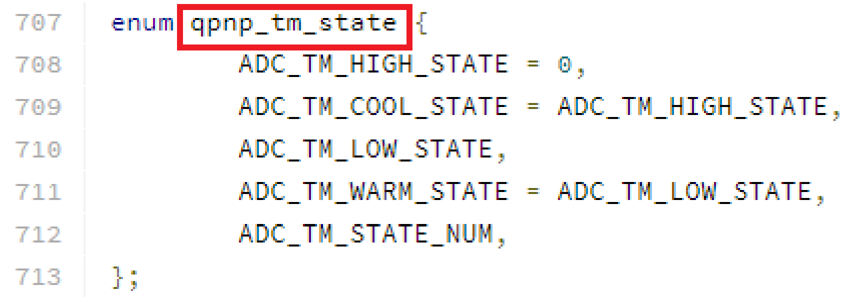
https://www.qualcomm.com/products/snapdragon-835-mobile-platform

	 **Snapdragon 835 mobile platform advancements:** + Snapdragon X16 LTE modem: mobile connectivity with LTE download speeds up to 1 Gbps, multi-gigabit 802.11ad, and integrated 2x2 802.11ac Wi-Fi with MU-MIMO + Qualcomm® Quick Charge™ 4 technology: 20% faster, 30% more efficient than our previous generation, charge from zero to up to 50% in 15 minutes[2] + Qualcomm® Adreno™ 540 GPU with visual processing subsystem: Advanced 3-D graphics rendering and up to 60X more colors help deliver life-like visuals for immersive experiences[1] + Qualcomm Spectra™ 180 Camera ISP: Dual 14-bit ISPs support up to 32MP single or dual 16MP cameras for the ultimate photography and videography experience + Qualcomm® Hexagon™ 682 DSP: Support for latest Machine Learning frameworks and image processing. Includes Hexagon Vector eXtensions and Qualcomm All-Ways Aware™ technology utilizing connectivity and sensors https://www.qualcomm.com/media/documents/files/snapdragon-835-mobile-platform-product-brief.pdf

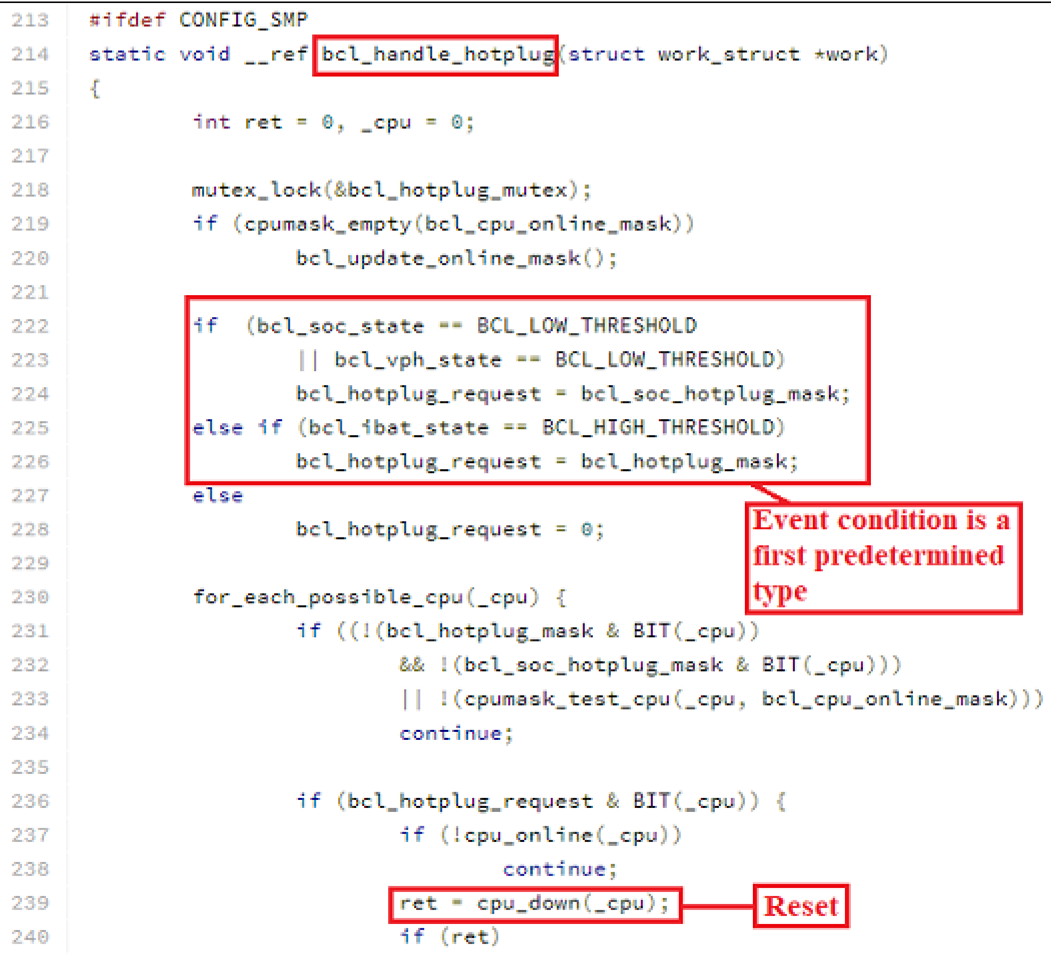


```
        qcom,bcl {
            compatible = "qcom,bcl";
            qcom,bcl-enable;
            qcom,bcl-framework-interface;
            qcom,bcl-freq-control-list = <0x1a 0x1b 0x1c 0x1d>;
            qcom,bcl-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
            qcom,bcl-soc-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;

            qcom,ibat-monitor {
                qcom,low-threshold-uamp = <0x33e140>;
                qcom,high-threshold-uamp = <0x401640>;
                qcom,mitigation-freq-khz = <0x8ca00>;
                qcom,vph-high-threshold-uv = <0x3567e0>;
                qcom,vph-low-threshold-uv = <0x325aa0>;
                qcom,soc-low-threshold = <0xa>;
                qcom,thermal-handle = <0xa0>;
            };
        };
```

https://pastebin.com/U0i7nP4P

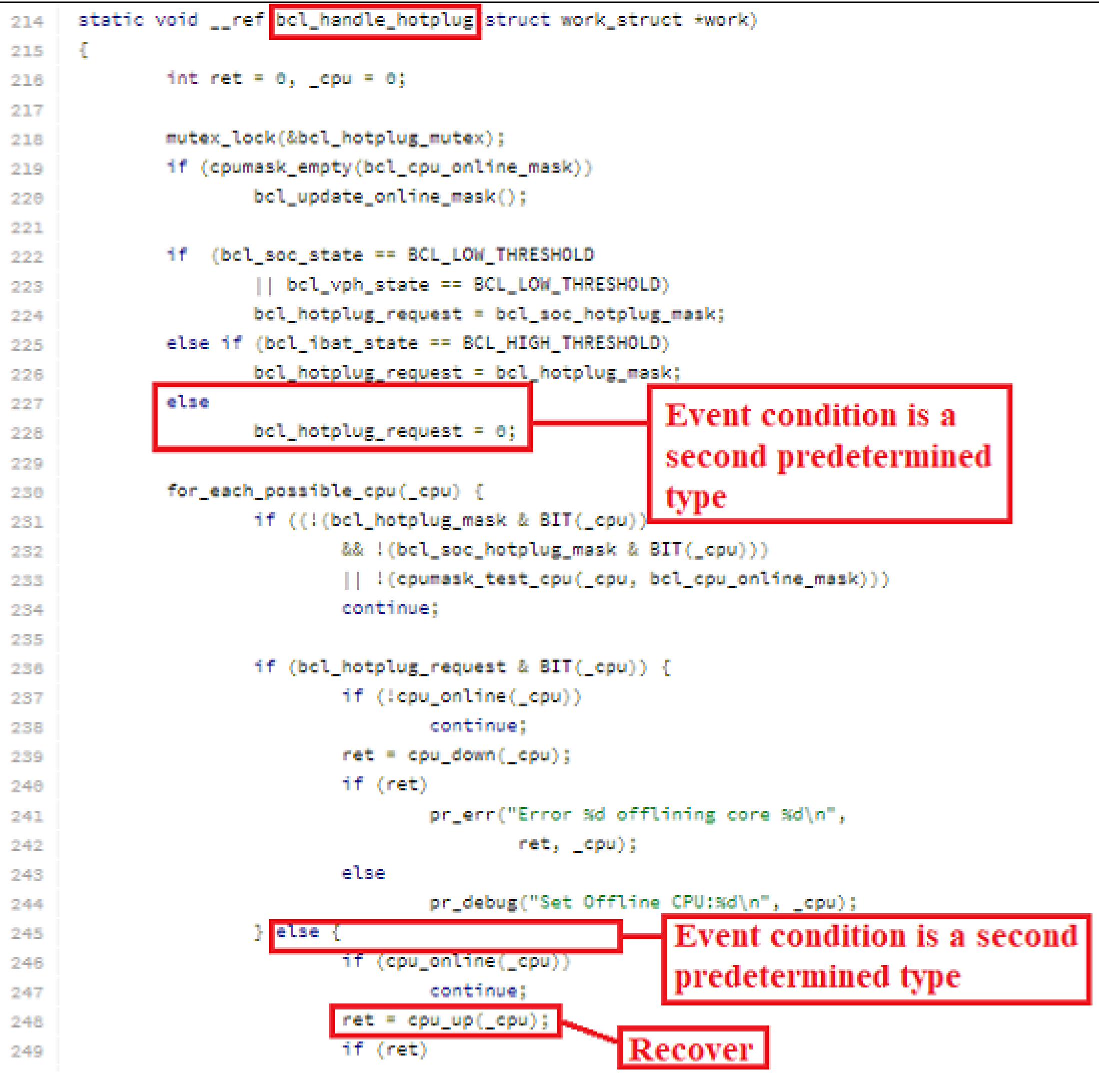
	 `bcl->btm_vph_adc_param.btm_ctx = bcl;`
	`bcl->btm_vph_adc_param.threshold_notification = bcl_vph_notification;`
	`bcl->btm_vph_adc_param.channel = bcl->btm_vph_chan;`
	
	`bcl->btm_ibat_adc_param.btm_ctx = bcl;`
	`bcl->btm_ibat_adc_param.threshold_notification = bcl_ibat_notification;`
	`bcl->btm_ibat_adc_param.channel = bcl->btm_ibat_chan;`
	
	`static void bcl_ibat_notification(enum qpnp_tm_state state, void *ctx);`
	`static void bcl_vph_notification(enum qpnp_tm_state state, void *ctx);`
	
	https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c
	 `enum qpnp_tm_state {`
	`        ADC_TM_HIGH_STATE = 0,`
	`        ADC_TM_COOL_STATE = ADC_TM_HIGH_STATE,`
	`        ADC_TM_LOW_STATE,`
	`        ADC_TM_WARM_STATE = ADC_TM_LOW_STATE,`
	`        ADC_TM_STATE_NUM,`
	`};`
	https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-asus-3.10-nougat-mr1-wear-release/include/linux/qpnp/qpnp-adc.h

(A) detecting an event;	The accused product discloses detecting an event (e.g., detecting if state is high or low).



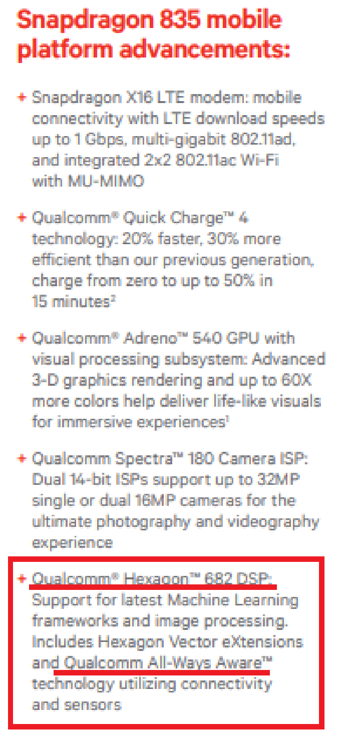
```
#ifdef CONFIG_SMP
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
```

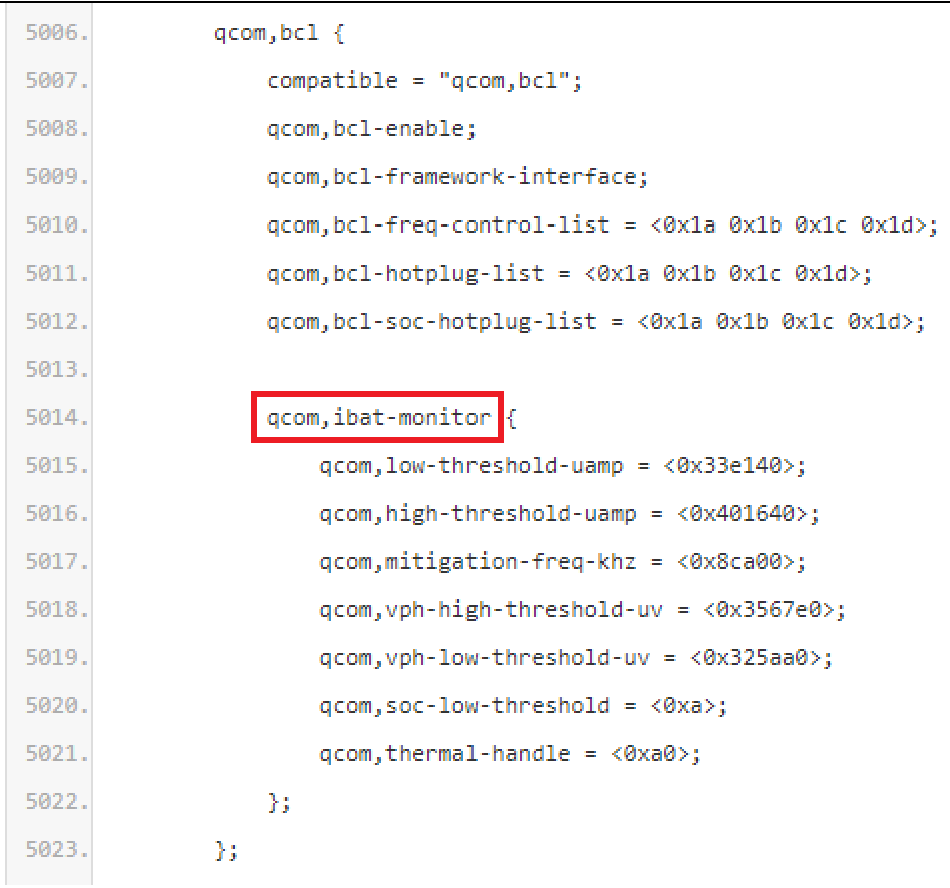
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c


Event condition is a second predetermined type
Event condition is a second predetermined type
Recover

```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
                                pr_err("Error %d offlining core %d\n",
                                        ret, _cpu);
                        else
                                pr_debug("Set Offline CPU:%d\n", _cpu);
                } else {
                        if (cpu_online(_cpu))
                                continue;
                        ret = cpu_up(_cpu);
                        if (ret)
```

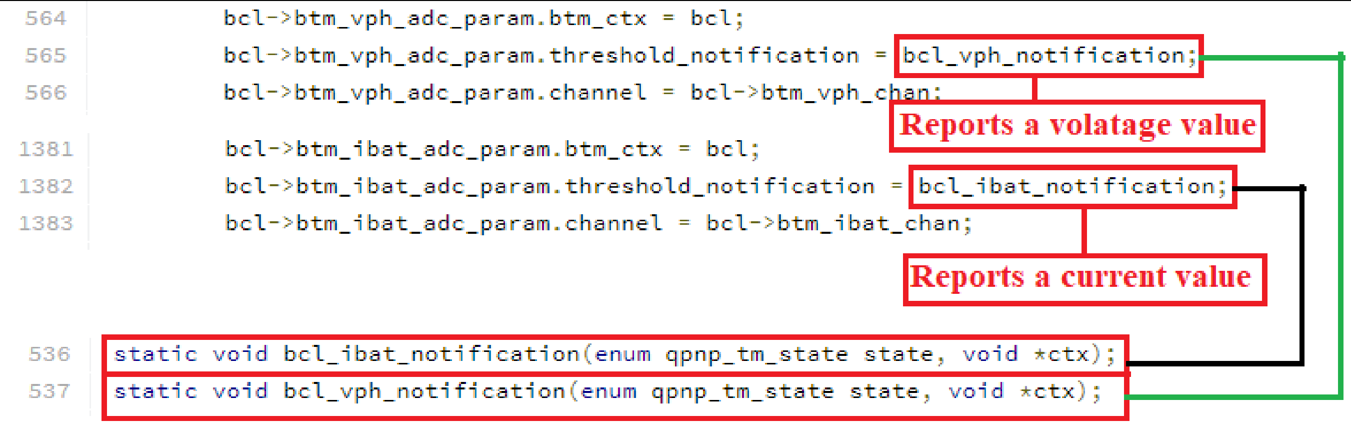
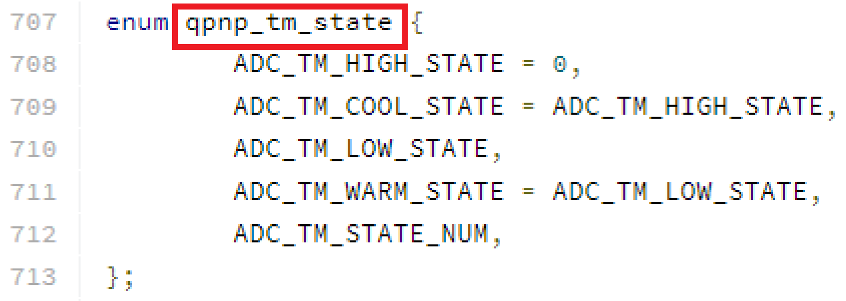
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

	As shown below, the Snapdragon 835 includes a battery monitoring circuit that generates a signal based upon the occurrence of a certain condition (in this case voltage variances from normal values). Qualcomm snapdragon Snapdragon 835 Mobile Platform https://www.qualcomm.com/products/snapdragon-835-mobile-platform  **Snapdragon 835 mobile platform advancements:** + Snapdragon X16 LTE modem: mobile connectivity with LTE download speeds up to 1 Gbps, multi-gigabit 802.11ad, and integrated 2x2 802.11ac Wi-Fi with MU-MIMO + Qualcomm® Quick Charge™ 4 technology: 20% faster, 30% more efficient than our previous generation, charge from zero to up to 50% in 15 minutes[2] + Qualcomm® Adreno™ 540 GPU with visual processing subsystem: Advanced 3-D graphics rendering and up to 60X more colors help deliver life-like visuals for immersive experiences[1] + Qualcomm Spectra™ 180 Camera ISP: Dual 14-bit ISPs support up to 32MP single or dual 16MP cameras for the ultimate photography and videography experience + Qualcomm® Hexagon™ 682 DSP: Support for latest Machine Learning frameworks and image processing. Includes Hexagon Vector eXtensions and Qualcomm All-Ways Aware™ technology utilizing connectivity and sensors https://www.qualcomm.com/media/documents/files/snapdragon-835-mobile-platform-product-brief.pdf

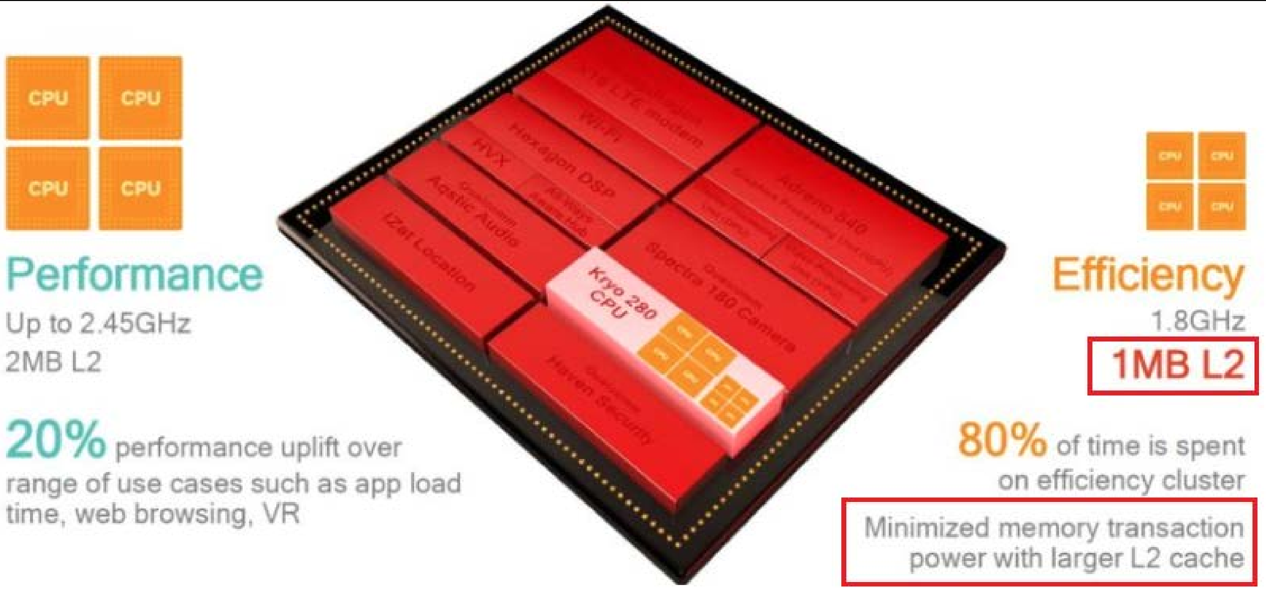


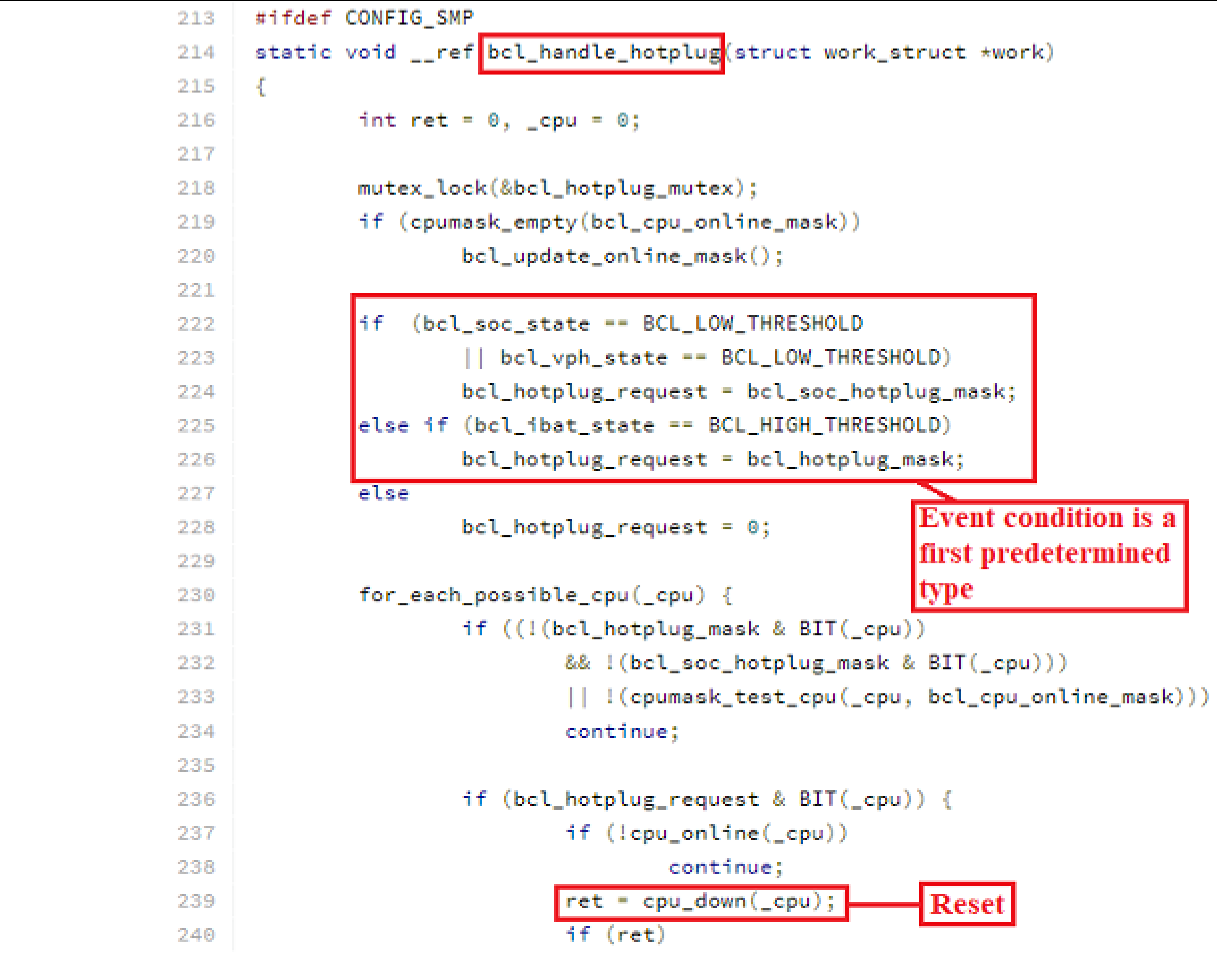
```
        qcom,bcl {
            compatible = "qcom,bcl";
            qcom,bcl-enable;
            qcom,bcl-framework-interface;
            qcom,bcl-freq-control-list = <0x1a 0x1b 0x1c 0x1d>;
            qcom,bcl-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;
            qcom,bcl-soc-hotplug-list = <0x1a 0x1b 0x1c 0x1d>;

            qcom,ibat-monitor {
                qcom,low-threshold-uamp = <0x33e140>;
                qcom,high-threshold-uamp = <0x401640>;
                qcom,mitigation-freq-khz = <0x8ca00>;
                qcom,vph-high-threshold-uv = <0x3567e0>;
                qcom,vph-low-threshold-uv = <0x325aa0>;
                qcom,soc-low-threshold = <0xa>;
                qcom,thermal-handle = <0xa0>;
            };
        };
```

https://pastebin.com/U0i7nP4P

	 `bcl->btm_vph_adc_param.btm_ctx = bcl;`
	`bcl->btm_vph_adc_param.threshold_notification = bcl_vph_notification;`
	`bcl->btm_vph_adc_param.channel = bcl->btm_vph_chan;`
	
	`bcl->btm_ibat_adc_param.btm_ctx = bcl;`
	`bcl->btm_ibat_adc_param.threshold_notification = bcl_ibat_notification;`
	`bcl->btm_ibat_adc_param.channel = bcl->btm_ibat_chan;`
	
	`static void bcl_ibat_notification(enum qpnp_tm_state state, void *ctx);`
	`static void bcl_vph_notification(enum qpnp_tm_state state, void *ctx);`
	
	https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c
	 `enum qpnp_tm_state {`
	`        ADC_TM_HIGH_STATE = 0,`
	`        ADC_TM_COOL_STATE = ADC_TM_HIGH_STATE,`
	`        ADC_TM_LOW_STATE,`
	`        ADC_TM_WARM_STATE = ADC_TM_LOW_STATE,`
	`        ADC_TM_STATE_NUM,`
	`};`
	https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-asus-3.10-nougat-mr1-wear-release/include/linux/qpnp/qpnp-adc.h

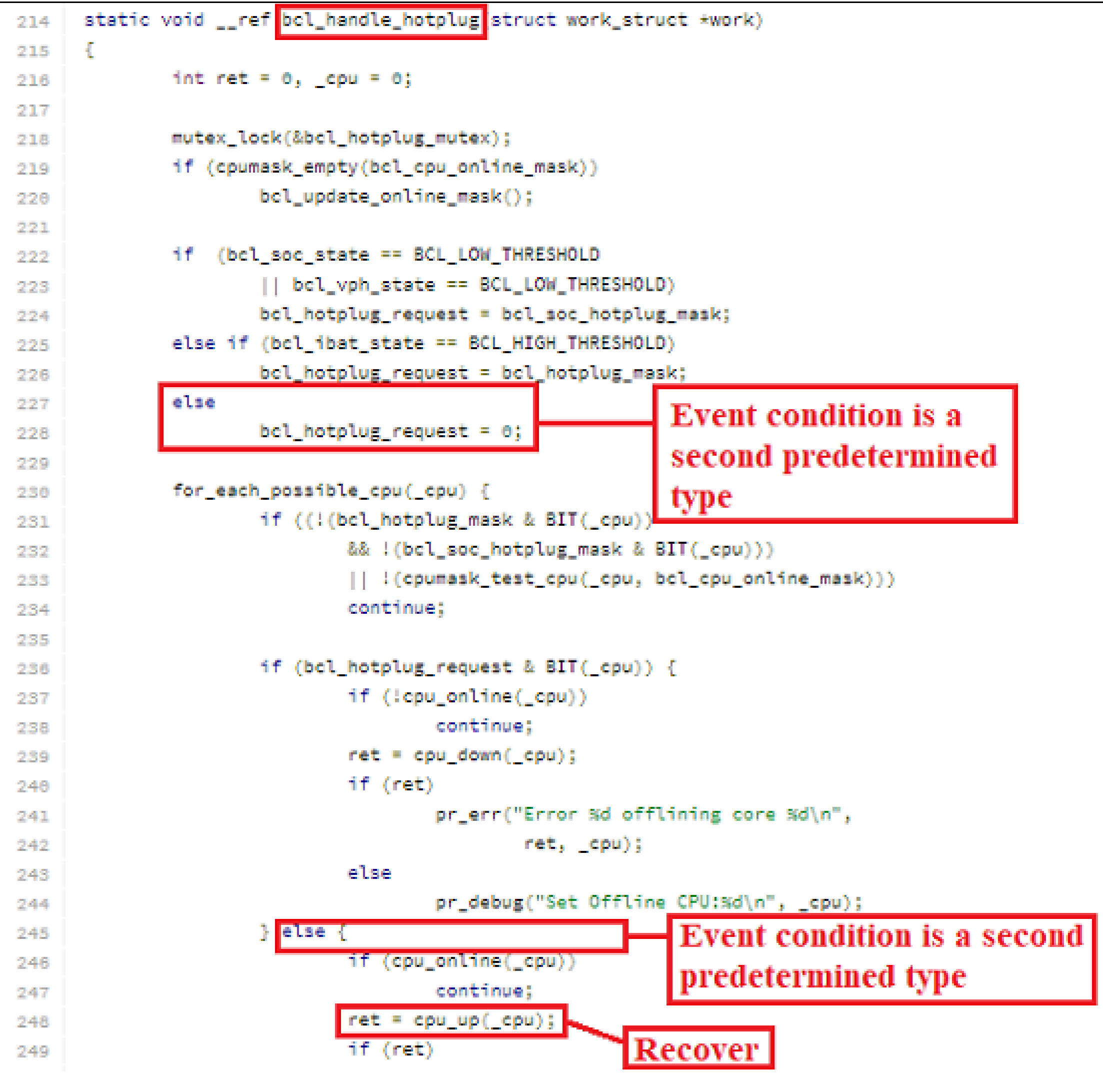
(B) storing said event;	The accused product discloses storing (e.g., storing in L2 cache) said event (e.g., if state is high or low). As shown below, the Snapdragon 835 includes an L2 cache that stores voltage variance events.

	 CPU CPU CPU CPU Performance Up to 2.45GHz 2MB L2 20% performance uplift over range of use cases such as app load time, web browsing, VR X16 LTE modem Wi-Fi Hexagon DSP HVX Adreno 540 Aqstic Audio IZat Location Kryo 280 CPU Spectra 180 Camera Haven Security CPU CPU CPU CPU Efficiency 1.8GHz 1MB L2 80% of time is spent on efficiency cluster Minimized memory transaction power with larger L2 cache https://www.androidauthority.com/qualcomm-details-snapdragon-835-735688/



```
#ifdef CONFIG_SMP
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
```

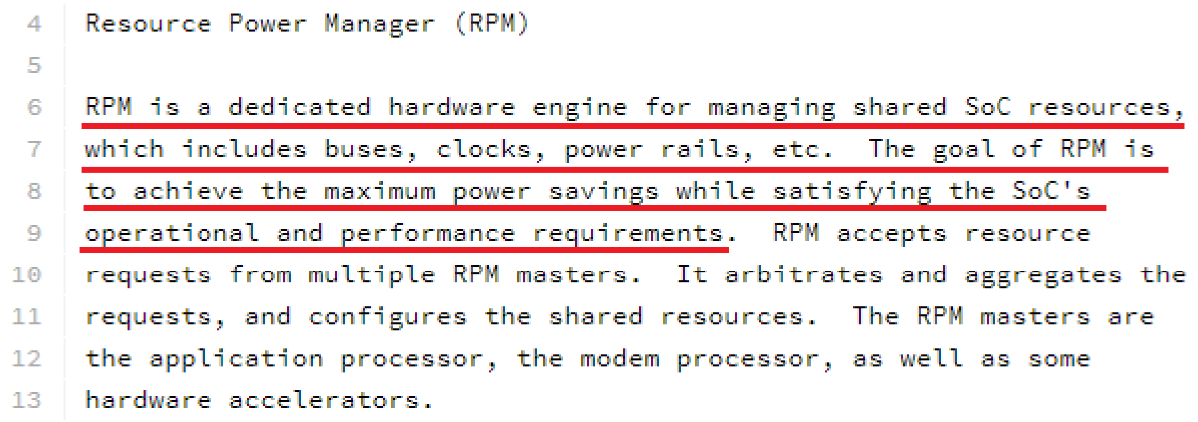
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

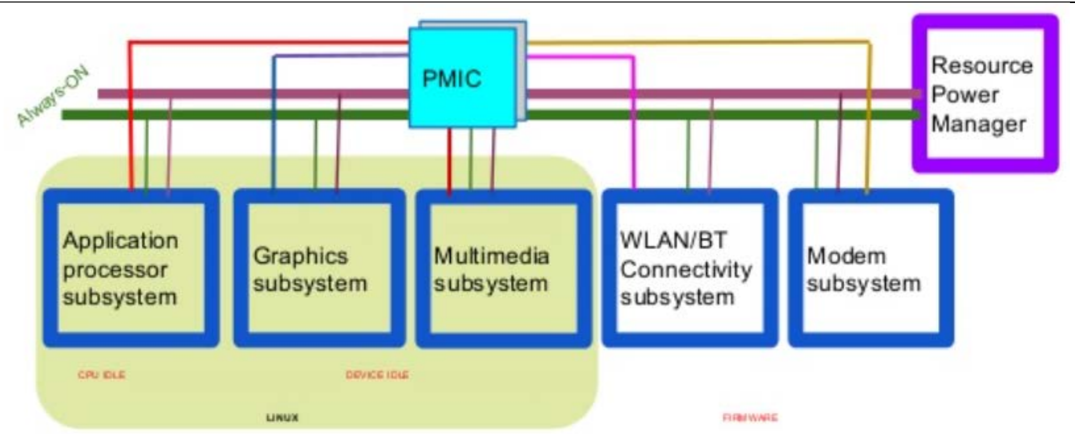


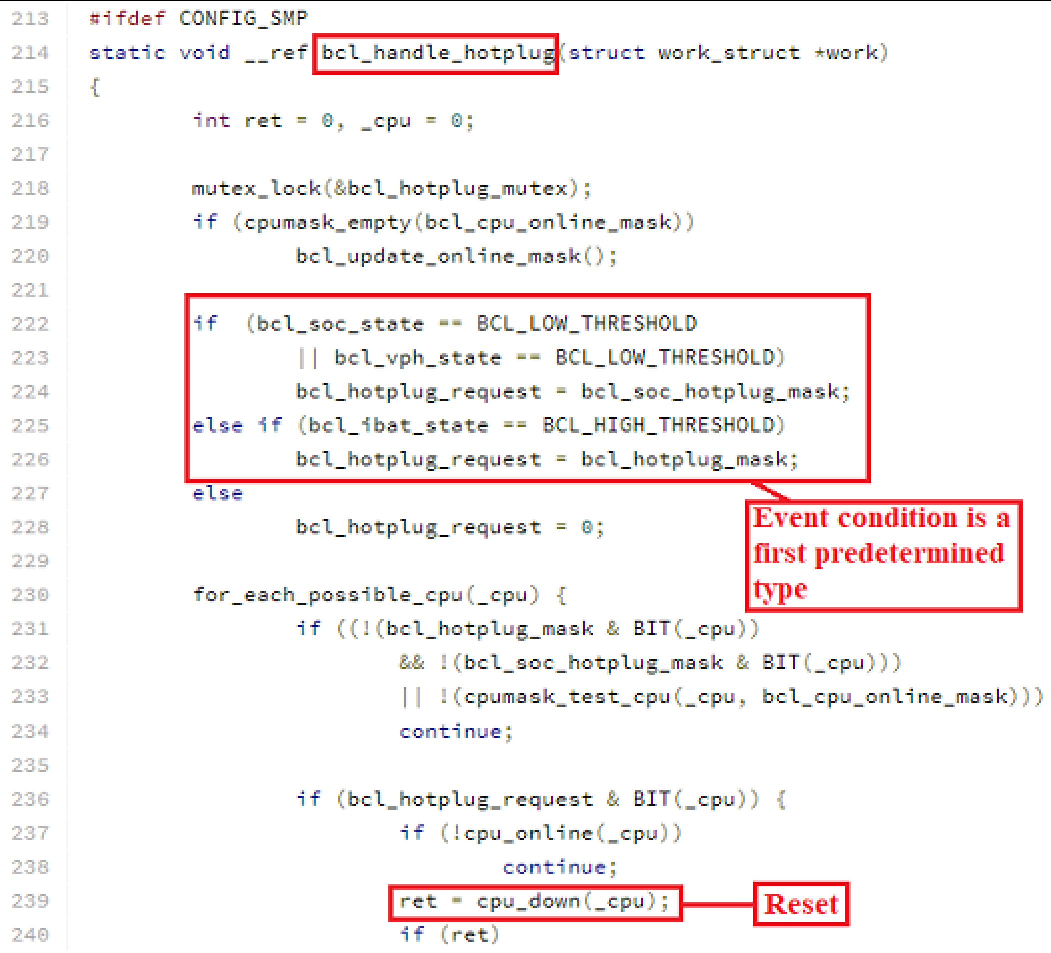
```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
                                pr_err("Error %d offlining core %d\n",
                                        ret, _cpu);
                        else
                                pr_debug("Set Offline CPU:%d\n", _cpu);
                } else {
                        if (cpu_online(_cpu))
                                continue;
                        ret = cpu_up(_cpu);
                        if (ret)
```

Event condition is a second predetermined type
Event condition is a second predetermined type
Recover

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

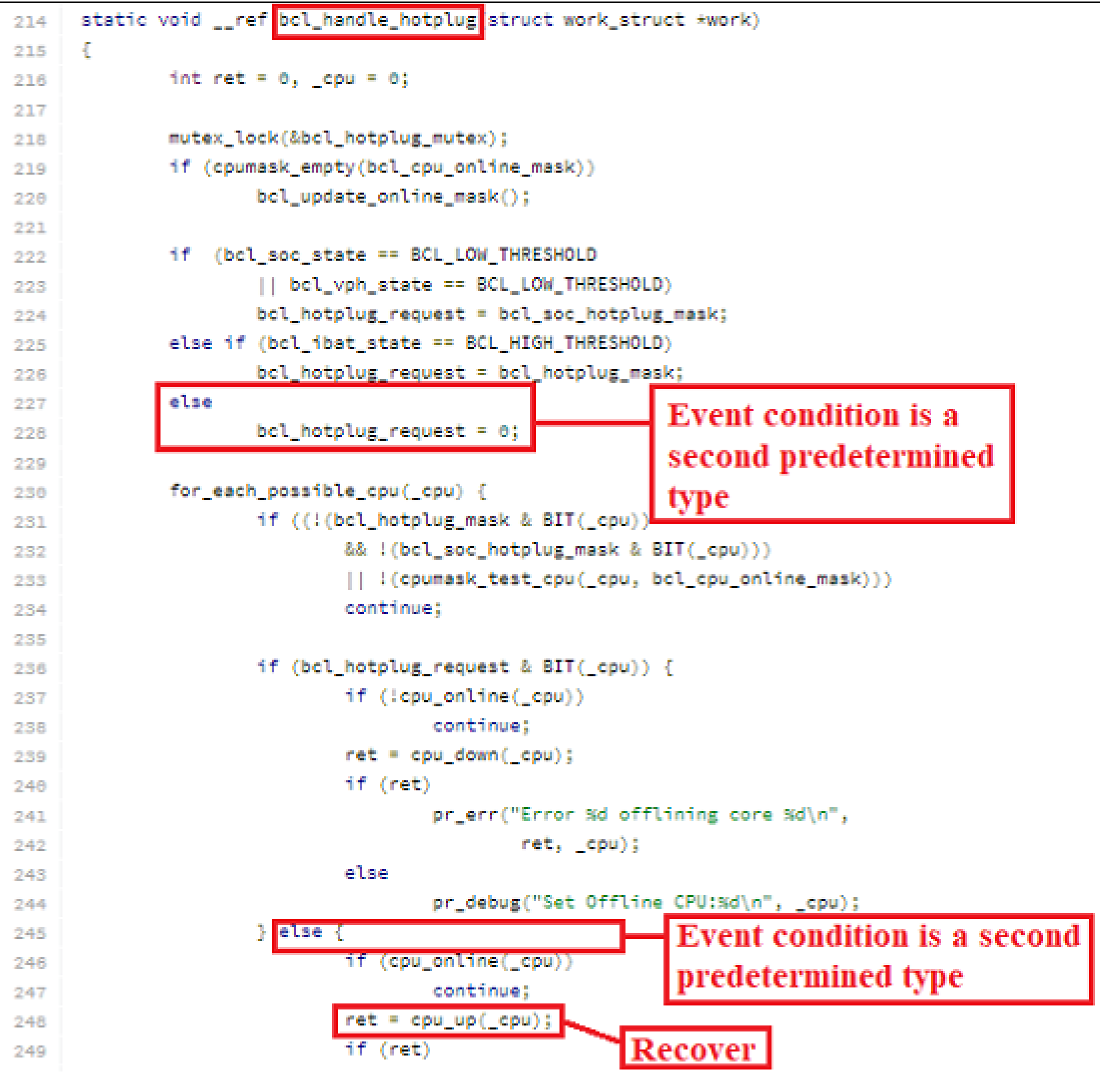
(C) comparing said stored event to a plurality of event types stored in a table to determine if said event is a first predetermined type or a second predetermined type; and	The accused product discloses comparing said stored event to a plurality of event types (e.g., the comparison of collected values with stored thresholds) stored in a table (e.g., a table containing various thresholds) to determine if said event is a first predetermined type (e.g., when bcl_soc_state == BCL_LOW_THRESHOLD OR bcl_vph_state == BCL_LOW_THRESHOLD) or a second predetermined type (e.g., when bcl_soc_state is not equal to BCL_LOW_THRESHOLD, bcl_vph_state is not equal to BCL_LOW_THRESHOLD and bcl_ibat_state is not equal to BCL_HIGH_THRESHOLD). 
	```
	Resource Power Manager (RPM)

	RPM is a dedicated hardware engine for managing shared SoC resources,
	which includes buses, clocks, power rails, etc.  The goal of RPM is
	to achieve the maximum power savings while satisfying the SoC's
	operational and performance requirements.  RPM accepts resource
	requests from multiple RPM masters.  It arbitrates and aggregates the
	requests, and configures the shared resources.  The RPM masters are
	the application processor, the modem processor, as well as some
	hardware accelerators.
	```
	 https://android.googlesource.com/kernel/msm/+/android-7.1.0_r0.2/Documentation/arm/msm/rpm.txt

	 Always-ON PMIC Resource Power Manager Application processor subsystem Graphics subsystem Multimedia subsystem WLAN/BT Connectivity subsystem Modem subsystem CPU IDLE DEVICE IDLE LINUX FIRMWARE https://www.slideshare.net/linaroorg/lcu14-210-qualcomm-snapdragon-power-management-unique-challenges-for-power-frameworks



```
#ifdef CONFIG_SMP
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
```

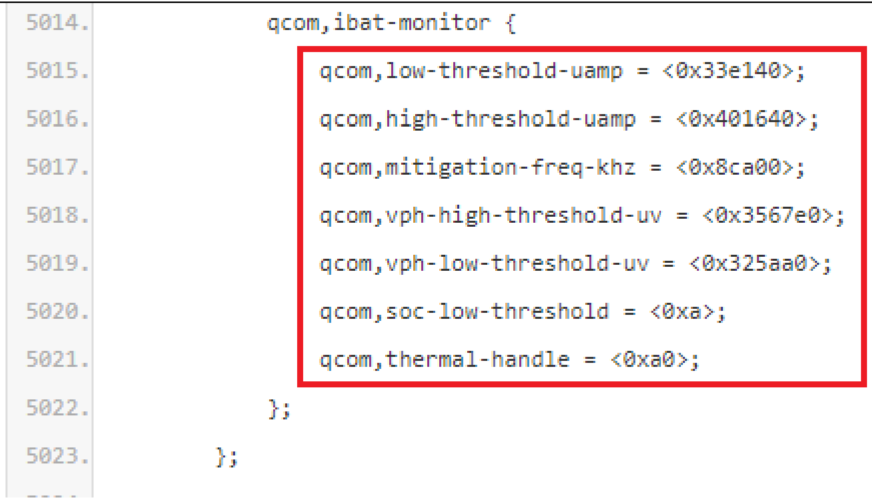
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c



```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
                                pr_err("Error %d offlining core %d\n",
                                        ret, _cpu);
                        else
                                pr_debug("Set Offline CPU:%d\n", _cpu);
                } else {
                        if (cpu_online(_cpu))
                                continue;
                        ret = cpu_up(_cpu);
                        if (ret)
```

Event condition is a second predetermined type
Event condition is a second predetermined type
Recover

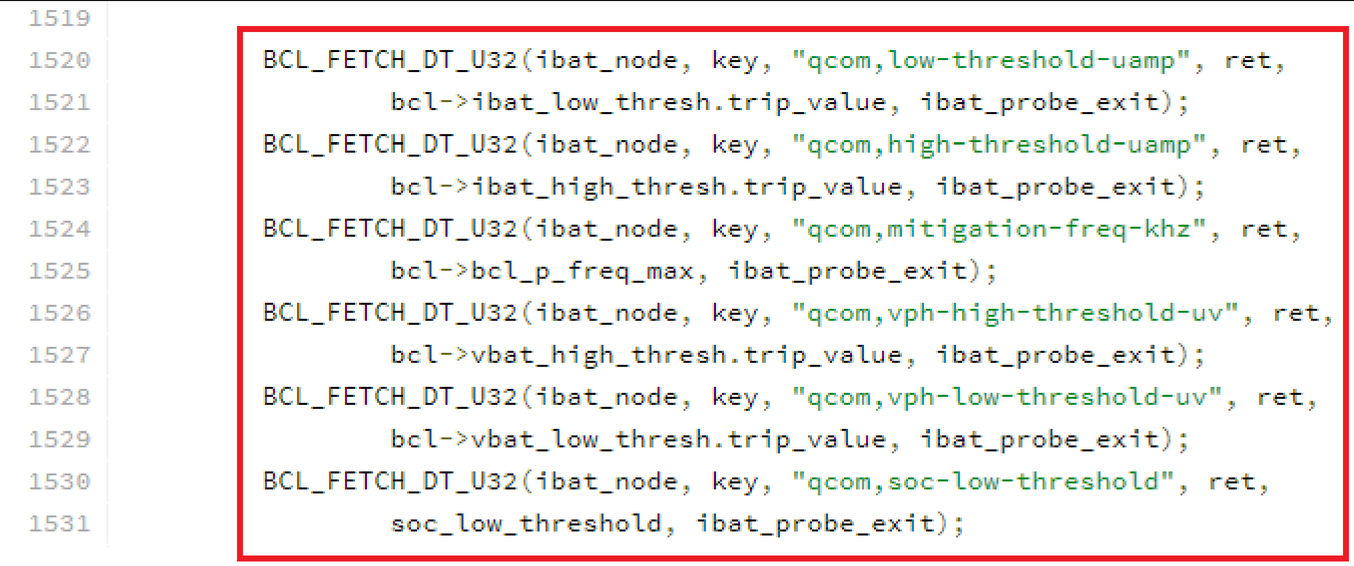
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c



```
            qcom,ibat-monitor {
                qcom,low-threshold-uamp = <0x33e140>;
                qcom,high-threshold-uamp = <0x401640>;
                qcom,mitigation-freq-khz = <0x8ca00>;
                qcom,vph-high-threshold-uv = <0x3567e0>;
                qcom,vph-low-threshold-uv = <0x325aa0>;
                qcom,soc-low-threshold = <0xa>;
                qcom,thermal-handle = <0xa0>;
            };
        };
```

https://pastebin.com/U0i7nP4P

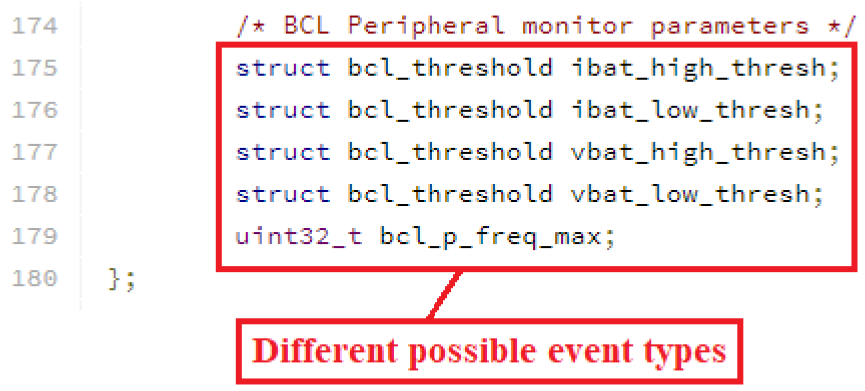
Threshold Values from the table (dtsi) are imported into the battery_current_limit module thru a record data type (bcl).



```
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,low-threshold-uamp", ret,
                bcl->ibat_low_thresh.trip_value, ibat_probe_exit);
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,high-threshold-uamp", ret,
                bcl->ibat_high_thresh.trip_value, ibat_probe_exit);
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,mitigation-freq-khz", ret,
                bcl->bcl_p_freq_max, ibat_probe_exit);
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,vph-high-threshold-uv", ret,
                bcl->vbat_high_thresh.trip_value, ibat_probe_exit);
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,vph-low-threshold-uv", ret,
                bcl->vbat_low_thresh.trip_value, ibat_probe_exit);
        BCL_FETCH_DT_U32(ibat_node, key, "qcom,soc-low-threshold", ret,
                soc_low_threshold, ibat_probe_exit);
```

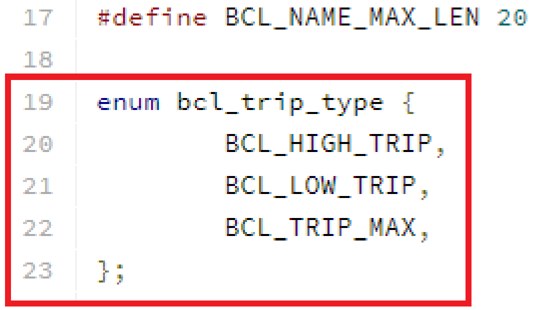
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

The values of the table are now inside the record, bcl. The State of Charge low threshold is saved in a variable soc_low_threshold.



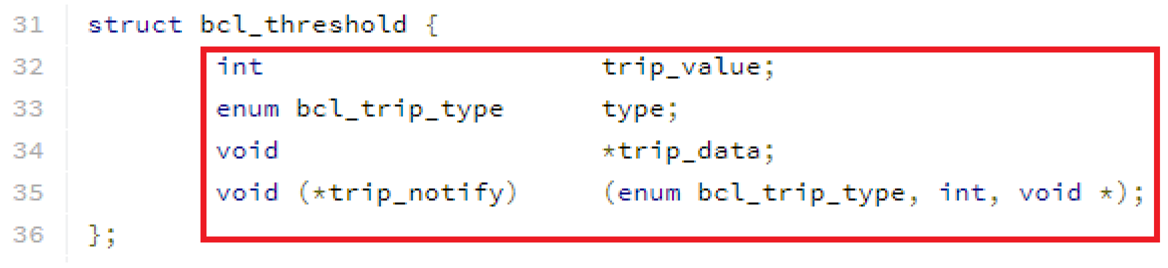
```
        /* BCL Peripheral monitor parameters */
        struct bcl_threshold ibat_high_thresh;
        struct bcl_threshold ibat_low_thresh;
        struct bcl_threshold vbat_high_thresh;
        struct bcl_threshold vbat_low_thresh;
        uint32_t bcl_p_freq_max;
};
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

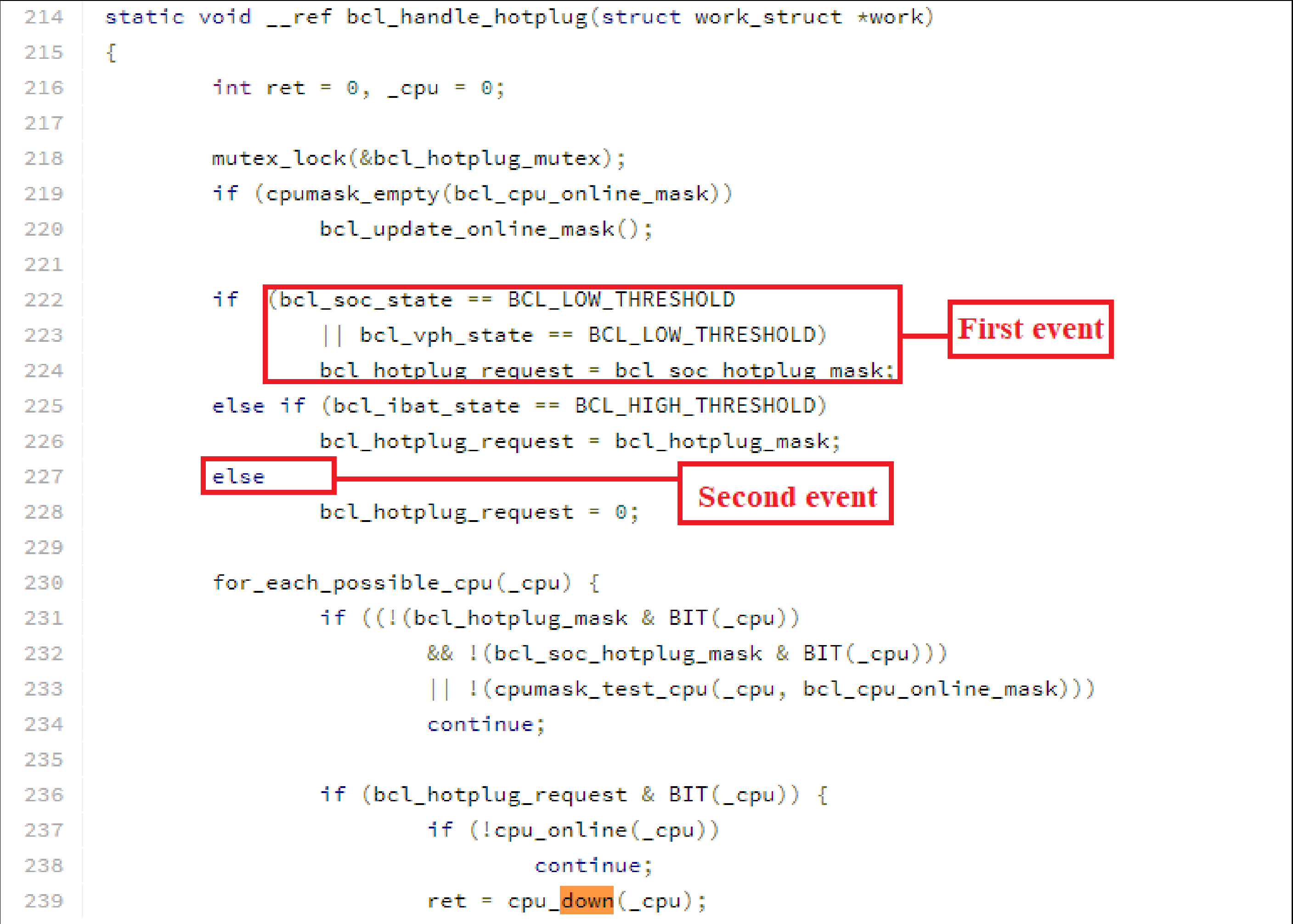


```
#define BCL_NAME_MAX_LEN 20

enum bcl_trip_type {
        BCL_HIGH_TRIP,
        BCL_LOW_TRIP,
        BCL_TRIP_MAX,
};
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/include/linux/msm_bcl.h



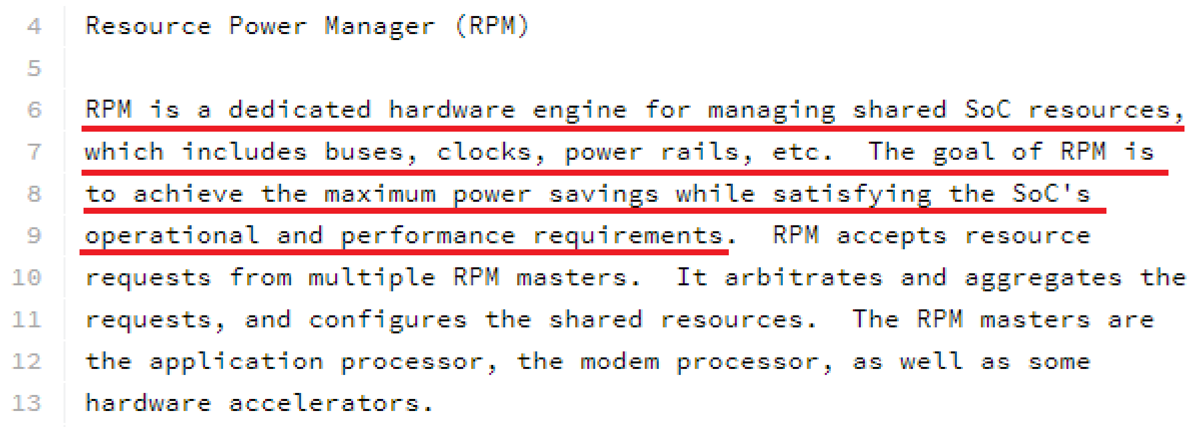
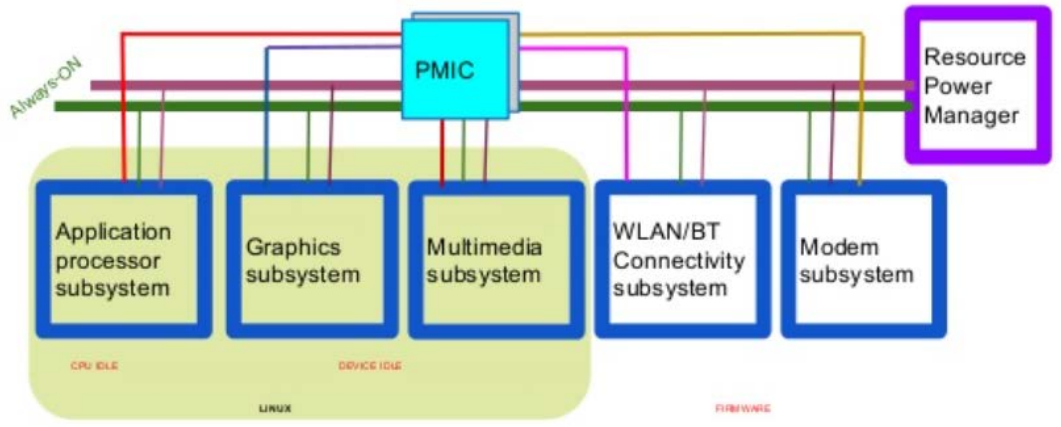
```
struct bcl_threshold {
        int                     trip_value;
        enum bcl_trip_type      type;
        void                    *trip_data;
        void (*trip_notify)     (enum bcl_trip_type, int, void *);
};
```

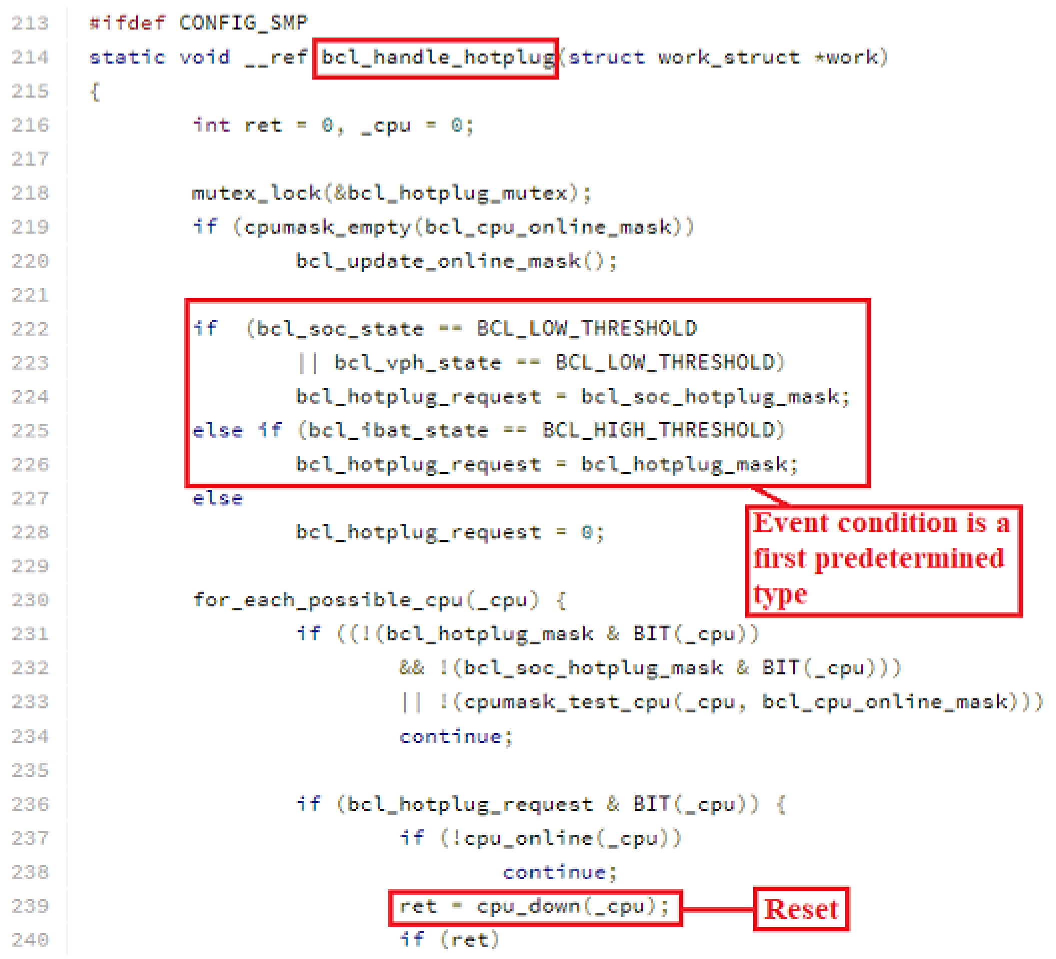


```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
```

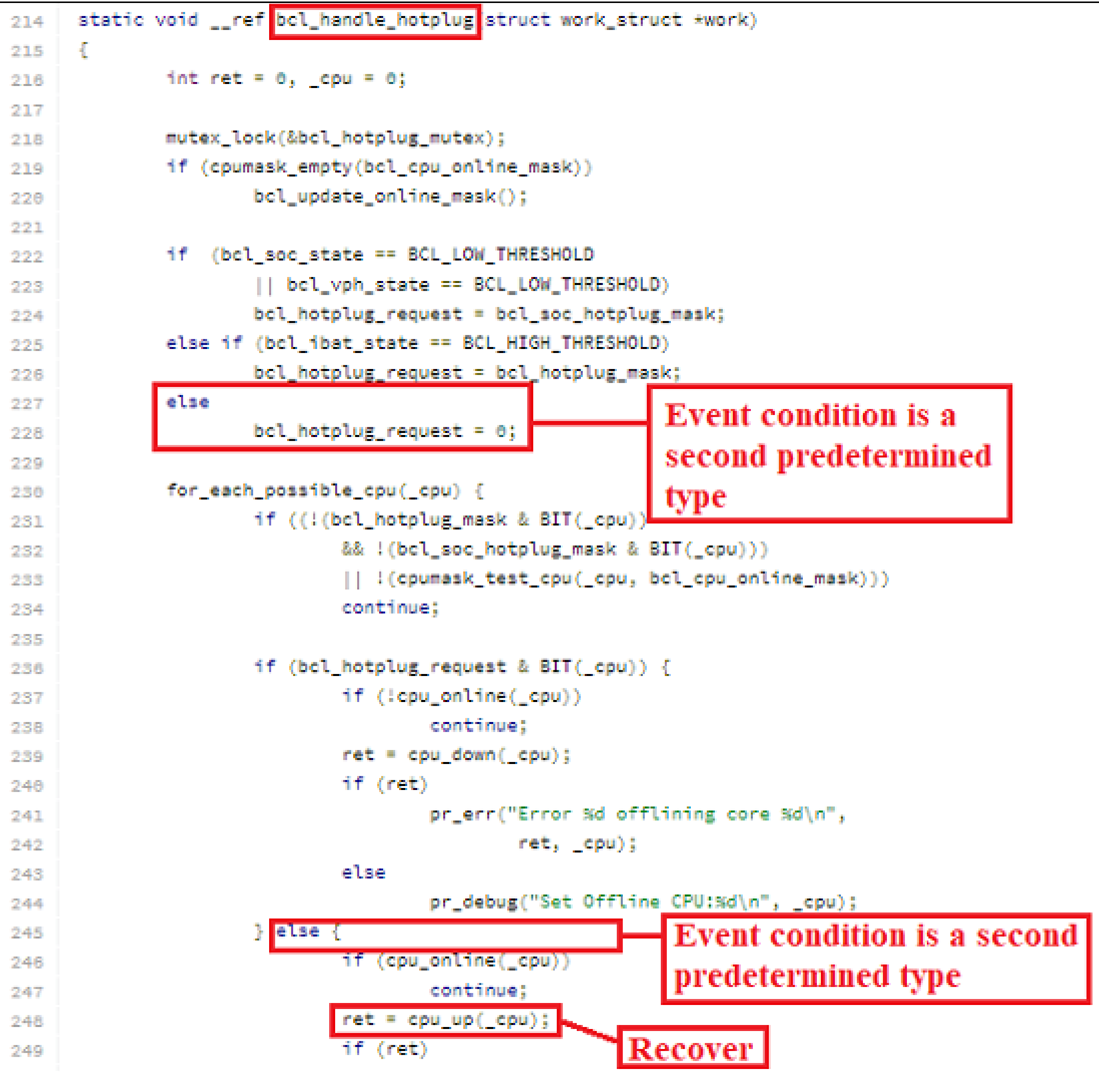
https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

The new values of bcl_vph_state and bcl_ibat_state are compared against the threshold values from the table.

(D) resetting a device when said event is a said first predetermined type and providing recovery when said event is a said second predetermined type.	The accused product discloses resetting (e.g., cpu_down) a device when said event is a said first predetermined type (e.g., when bcl_soc_state == BCL_LOW_THRESHOLD OR bcl_vph_state == BCL_LOW_THRESHOLD) and providing recovery (e.g., cpu_up) when said event is a said second predetermined type (e.g., when bcl_soc_state is not equal to BCL_LOW_THRESHOLD, bcl_vph_state is not equal to BCL_LOW_THRESHOLD and bcl_ibat_state is not equal to BCL_HIGH_THRESHOLD).  Resource Power Manager (RPM) RPM is a dedicated hardware engine for managing shared SoC resources, which includes buses, clocks, power rails, etc. The goal of RPM is to achieve the maximum power savings while satisfying the SoC's operational and performance requirements. RPM accepts resource requests from multiple RPM masters. It arbitrates and aggregates the requests, and configures the shared resources. The RPM masters are the application processor, the modem processor, as well as some hardware accelerators. https://android.googlesource.com/kernel/msm/+/android-7.1.0_r0.2/Documentation/arm/msm/rpm.txt  Always-ON PMIC Resource Power Manager Application processor subsystem Graphics subsystem Multimedia subsystem WLAN/BT Connectivity subsystem Modem subsystem CPU IDLE DEVICE IDLE LINUX FIRMWARE

	https://www.slideshare.net/linaroorg/lcu14-210-qualcomm-snapdragon-power-management-unique-challenges-for-power-frameworks  (code follows table) https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

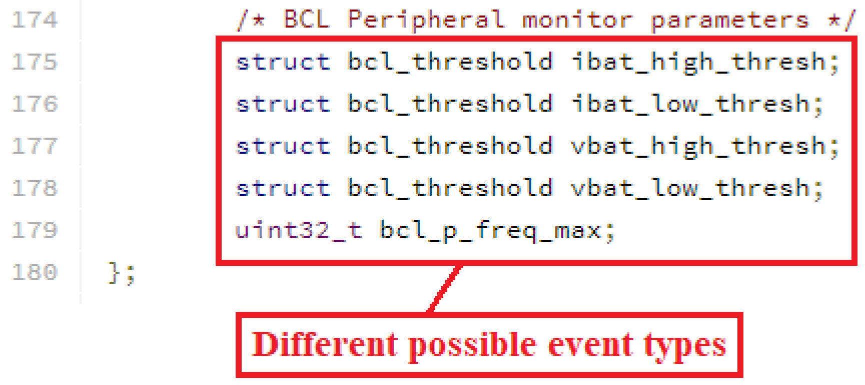
```
#ifdef CONFIG_SMP
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
	int ret = 0, _cpu = 0;

	mutex_lock(&bcl_hotplug_mutex);
	if (cpumask_empty(bcl_cpu_online_mask))
		bcl_update_online_mask();

	if  (bcl_soc_state == BCL_LOW_THRESHOLD
		|| bcl_vph_state == BCL_LOW_THRESHOLD)
		bcl_hotplug_request = bcl_soc_hotplug_mask;
	else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
		bcl_hotplug_request = bcl_hotplug_mask;
	else
		bcl_hotplug_request = 0;

	for_each_possible_cpu(_cpu) {
		if ((!(bcl_hotplug_mask & BIT(_cpu))
			&& !(bcl_soc_hotplug_mask & BIT(_cpu)))
			|| !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
			continue;

		if (bcl_hotplug_request & BIT(_cpu)) {
			if (!cpu_online(_cpu))
				continue;
			ret = cpu_down(_cpu);
			if (ret)
```



```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if  (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
                        if (ret)
                                pr_err("Error %d offlining core %d\n",
                                        ret, _cpu);
                        else
                                pr_debug("Set Offline CPU:%d\n", _cpu);
                } else {
                        if (cpu_online(_cpu))
                                continue;
                        ret = cpu_up(_cpu);
                        if (ret)
```

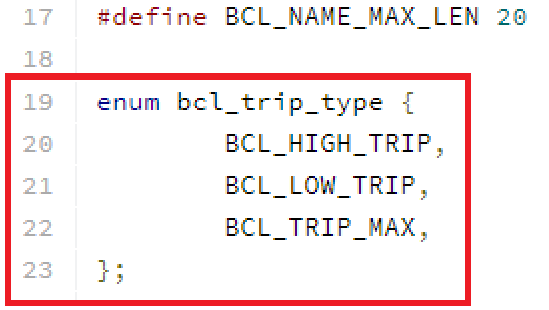
Event condition is a second predetermined type
Event condition is a second predetermined type
Recover

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c



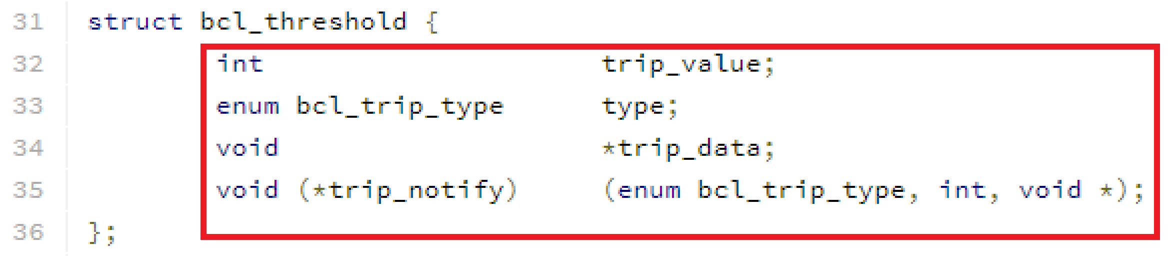
```
        /* BCL Peripheral monitor parameters */
        struct bcl_threshold ibat_high_thresh;
        struct bcl_threshold ibat_low_thresh;
        struct bcl_threshold vbat_high_thresh;
        struct bcl_threshold vbat_low_thresh;
        uint32_t bcl_p_freq_max;
};
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c

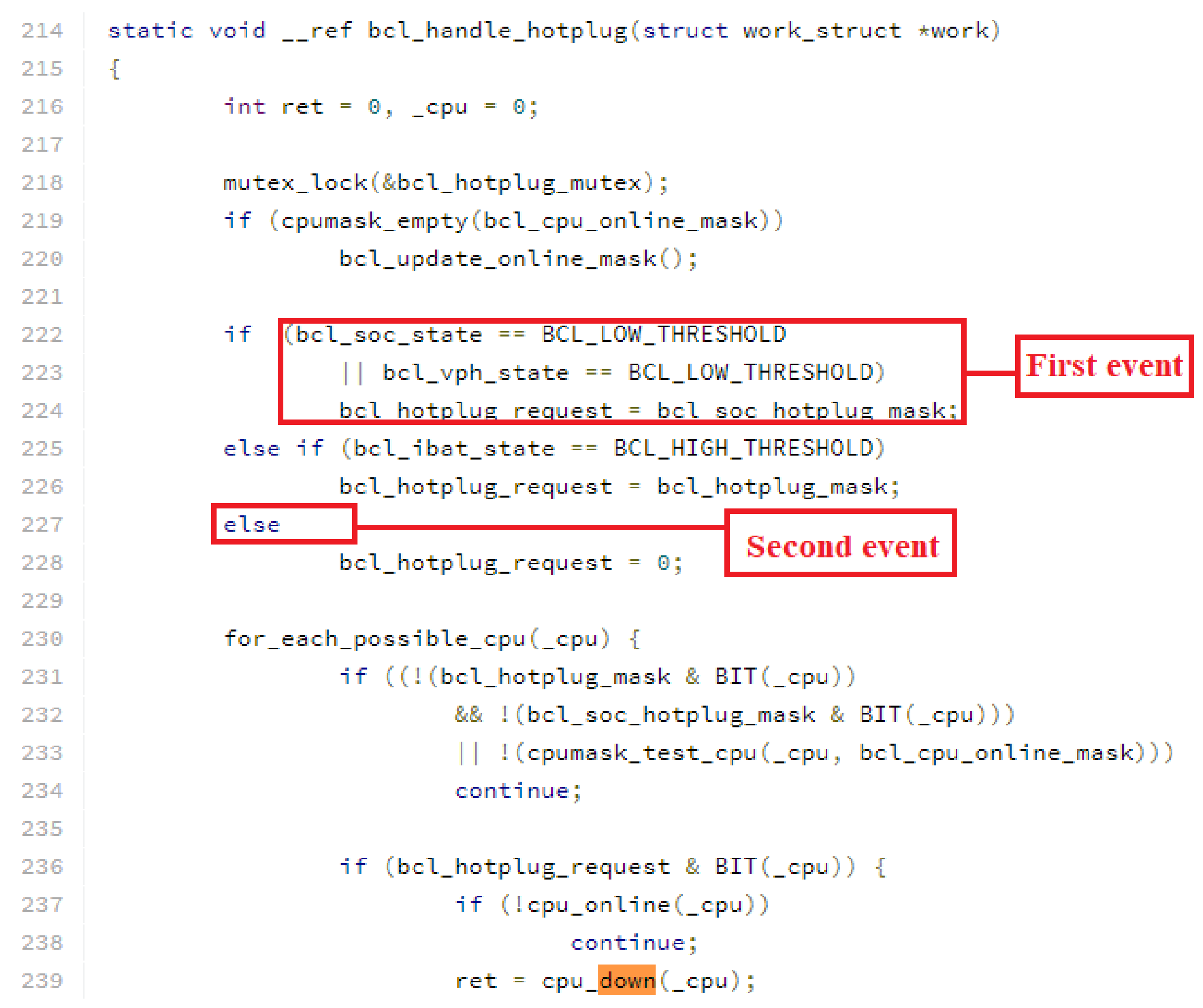


```
#define BCL_NAME_MAX_LEN 20

enum bcl_trip_type {
        BCL_HIGH_TRIP,
        BCL_LOW_TRIP,
        BCL_TRIP_MAX,
};
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/include/linux/msm_bcl.h



```
struct bcl_threshold {
        int                     trip_value;
        enum bcl_trip_type      type;
        void                    *trip_data;
        void (*trip_notify)     (enum bcl_trip_type, int, void *);
};
```



```
static void __ref bcl_handle_hotplug(struct work_struct *work)
{
        int ret = 0, _cpu = 0;

        mutex_lock(&bcl_hotplug_mutex);
        if (cpumask_empty(bcl_cpu_online_mask))
                bcl_update_online_mask();

        if (bcl_soc_state == BCL_LOW_THRESHOLD
                || bcl_vph_state == BCL_LOW_THRESHOLD)
                bcl_hotplug_request = bcl_soc_hotplug_mask;
        else if (bcl_ibat_state == BCL_HIGH_THRESHOLD)
                bcl_hotplug_request = bcl_hotplug_mask;
        else
                bcl_hotplug_request = 0;

        for_each_possible_cpu(_cpu) {
                if ((!(bcl_hotplug_mask & BIT(_cpu))
                        && !(bcl_soc_hotplug_mask & BIT(_cpu)))
                        || !(cpumask_test_cpu(_cpu, bcl_cpu_online_mask)))
                        continue;

                if (bcl_hotplug_request & BIT(_cpu)) {
                        if (!cpu_online(_cpu))
                                continue;
                        ret = cpu_down(_cpu);
```

https://android.googlesource.com/kernel/msm/+/refs/heads/android-msm-angler-3.10-nougat/drivers/power/battery_current_limit.c